Objects and Classes

Classes

- Classes are used to create objects
- ArrayList and HashMap are classes that are used to create objects
- How can we create our own classes that create objects?

```
ArrayList<Integer> arr1 = new ArrayList<>(Arrays.asList(10, 9, 8, 7));
```

Creating Our Own Classes and Objects

Classes

Let's create a Player class with 3 variables:

- Maximum hit points
- Current hit points
- Name

Player Class

```
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

}
```

- We create the Player class
- A class is a container for:
  - State - Variables that are stored in objects of this class
  - Behavior - All of the methods that can be called on that object

Player Class

```
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

}
```

- We'll declare variables outside of all the methods of the class
  - These are called "instance variables"
    - Also called "state variables", "fields", "object variables"
- A copy of these variables will be created for each object of type Player

Player Class

```
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

}
```

- We'll write a special method called a constructor
- The name of this method must be the name of the class
- No return type is specified

Player Class

```
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

}
```

- The constructor is the method that's called when we create a new Player object
- We create Player objects
  - new Player("name", 10);
- This constructor initializes our 3 instance variables

Player Class

```
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

}
```

- The keyword "this" stored a reference to the object that called a method
- The object that's being constructed when a constructor is called

Player Class

```
public class Player {
    private int maxHP = 10;
    private int hp = 10;
    private String name;

    public Player(String name) {
       this.name = name;
    }

}
```

- You can also initialize instance variables when they are declared
- Use this if you want every object to have the same initial value for a variable

Player Class

```
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

}
```

- Our instance variables are all private
  - Very common in Java
- Leverage encapsulation
  - Hide the details of your code
  - Expose public methods for others to interact with your code
- So how does anyone use this state?...

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

    public int getMaxHP() {
        return this.maxHP;
    }
    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }

    public int getHP() {
        return this.hp;
    }
    public void setHP(int hp) {
        this.hp = hp;
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

}
```

Player Class

- Getters and Setters!

- Write public methods that allow access to your state
- Getters - Return the value of the requested variable
- Setters - Takes a value and reassigned the instance variable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

    public int getMaxHP() {
        return this.maxHP;
    }
    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }

    public int getHP() {
        return this.hp;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

}
```

Player Class

- Why???
  - It would be easier to just make our variables public!
- Control.
  - If we want to sanitize values, add code to the setter
  - If you want to format output, add code to the getter
- If others write code to access your variables directly, you do not have this option!

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public int getMaxHP() {
        return this.maxHP;
    }
    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }

    public int getHP() {
        return this.hp;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

}
```

Player Class

- You can call your setters in your constructor
- Ensures your checks are ran when an object is created

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.maxHP = maxHP;
        this.hp = maxHP;
        this.name = name;
    }

    public int getMaxHP() {
        return this.maxHP;
    }
    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }

    public int getHP() {
        return this.hp;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

}
```

Player Class

- Notice that nothing in this class is static
- Use static if a method/variable should belong to the ***class***
- Do not use static if a method/variable should belong to an ***object*** created from the class

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public int getMaxHP() {
        return this.maxHP;
    }
    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }

    public int getHP() {
        return this.hp;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

}
```

Player Class

- Classes define new **types**
  - The ArrayList *class* defines the ArrayList *type*
  - Our Player *class* defines the Player *type*
- We can use Player wherever we could use any other type
  - As variable types
  - As parameter types in methods
  - As the return type of methods
  - As type parameters of data structures

Stack Memory

- Only primitive types are stored directly on the stack
  - double
  - int
  - char
  - boolean
  - String*
  - Double/Integer/Character/Boolean*

- **Everything else** is stored on the heap with only their references stored on the stack**
  - This includes **every** object created from a class that **you wrote**

***Strings and boxed types are actually more complex, but we will treat them as though they are on the stack in this course because they *behave* exactly as a value on the stack**

****Stack and heap allocations vary by compiler and JVM implementations. With modern optimizations, we can never be sure where our values will be stored**
We'll use this simplified view so we can move on and learn Computer Science

Memory Diagram

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap

in/out

- We'll trace this version of the code
- Set up the stack, heap, and in/out

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap

in/out

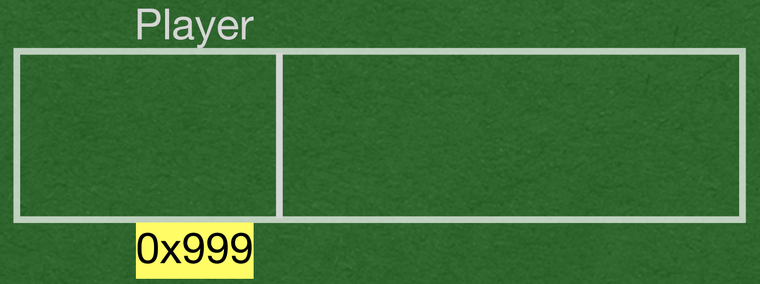
- New objects are created on the heap
- Only a reference is stored in variables

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap



in/out

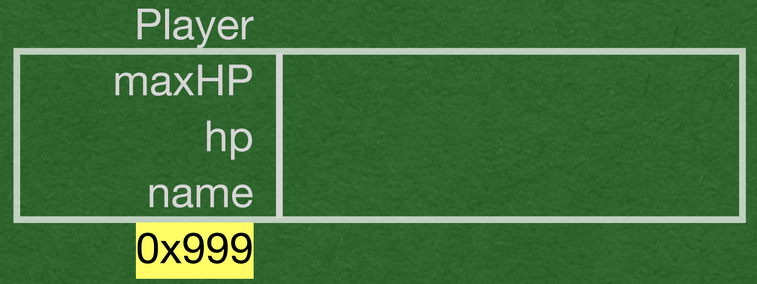
- Add the type of the object above it in the heap
- Make up a reference number and add it below the object

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap



in/out

- Create all the instance variables and add them to the object
- Each new object will have its own copies of each instance variable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
this	0x999
name	"Dark Cecil"
maxHP	10

Player

Heap

Player

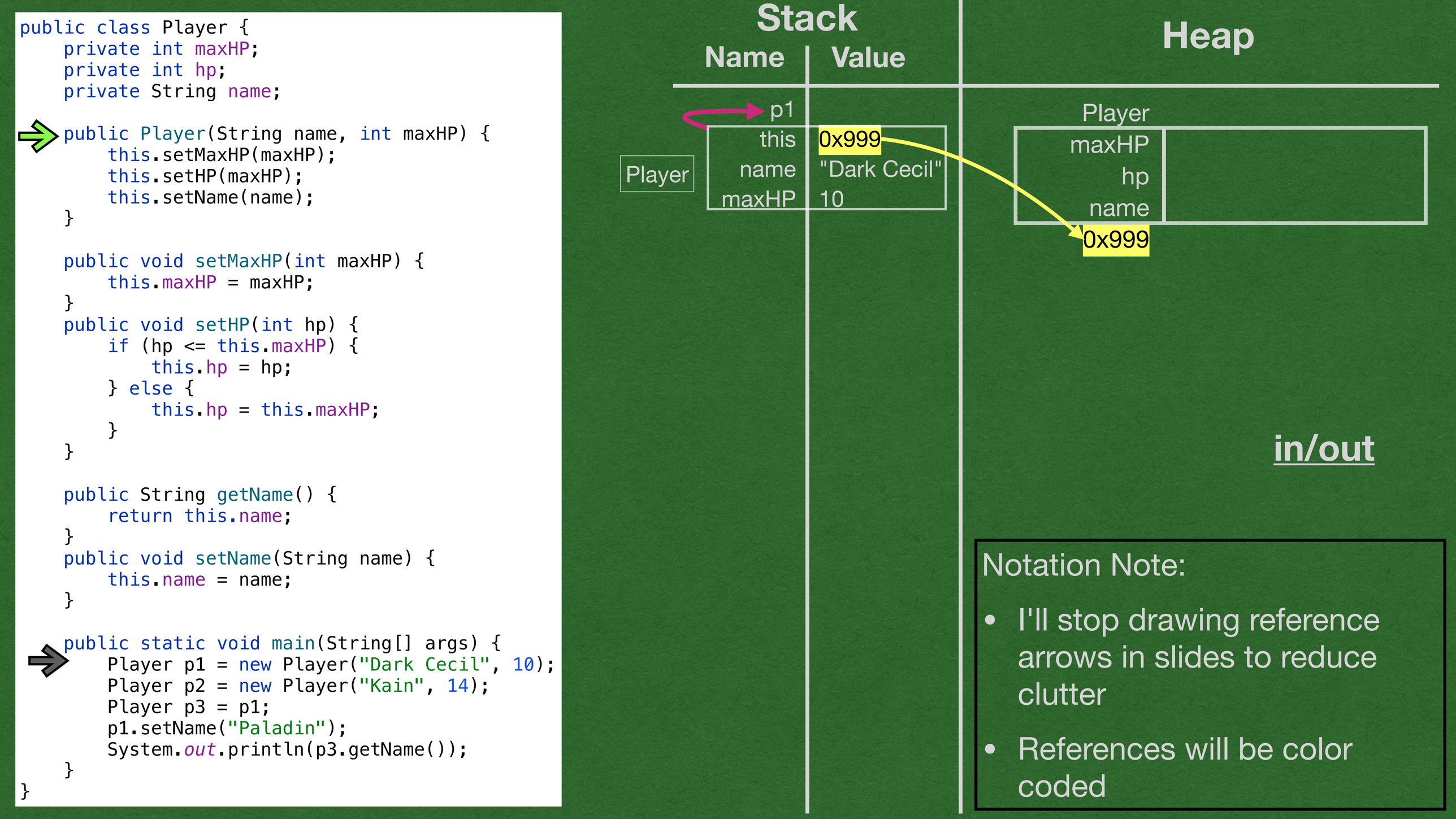
maxHP	
hp	
name	

0x999

in/out

- The constructor is a method
- Add it to the stack along with all of its parameters
- Constructors return a reference to the object that was created

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

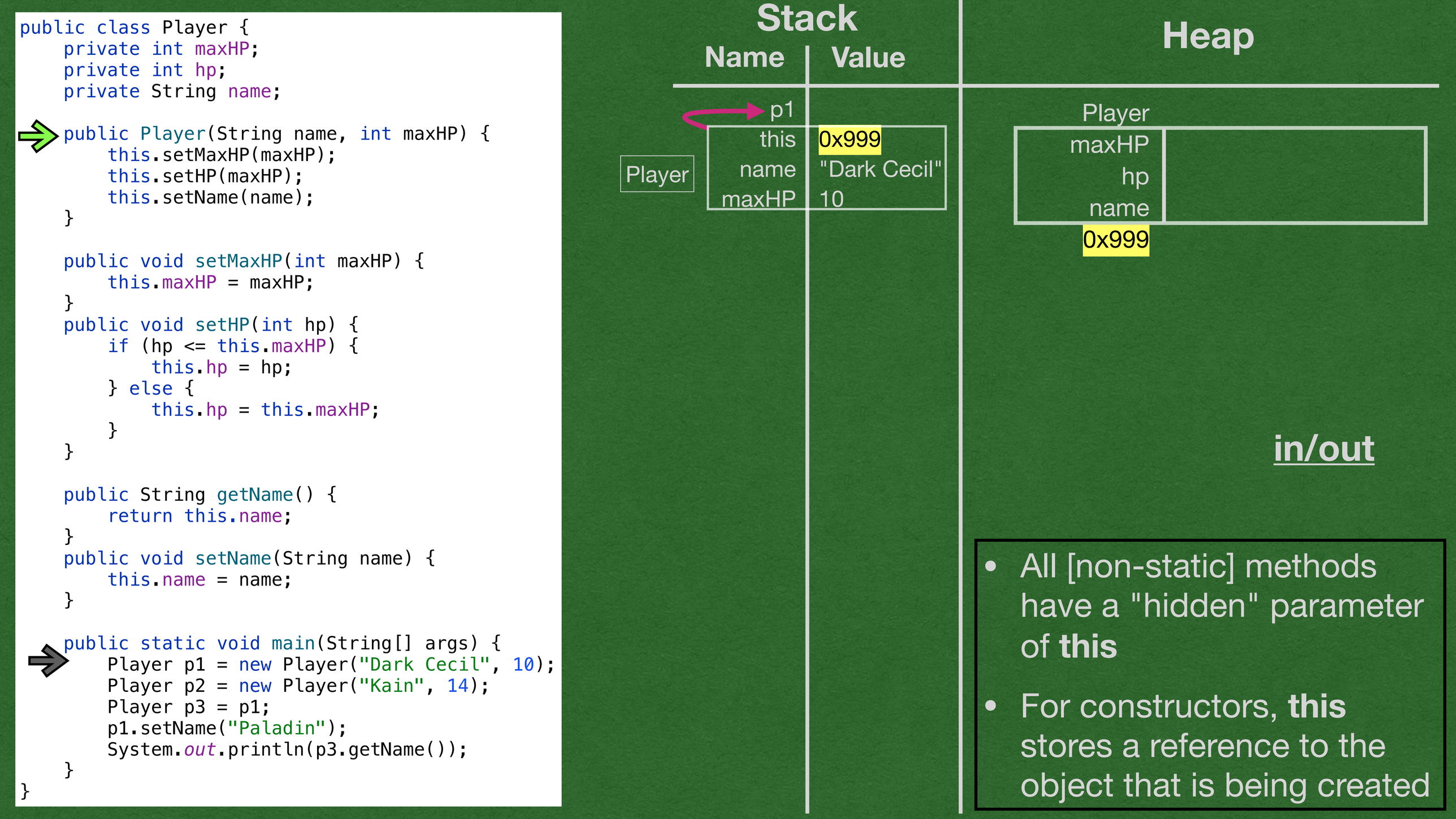


in/out

Notation Note:

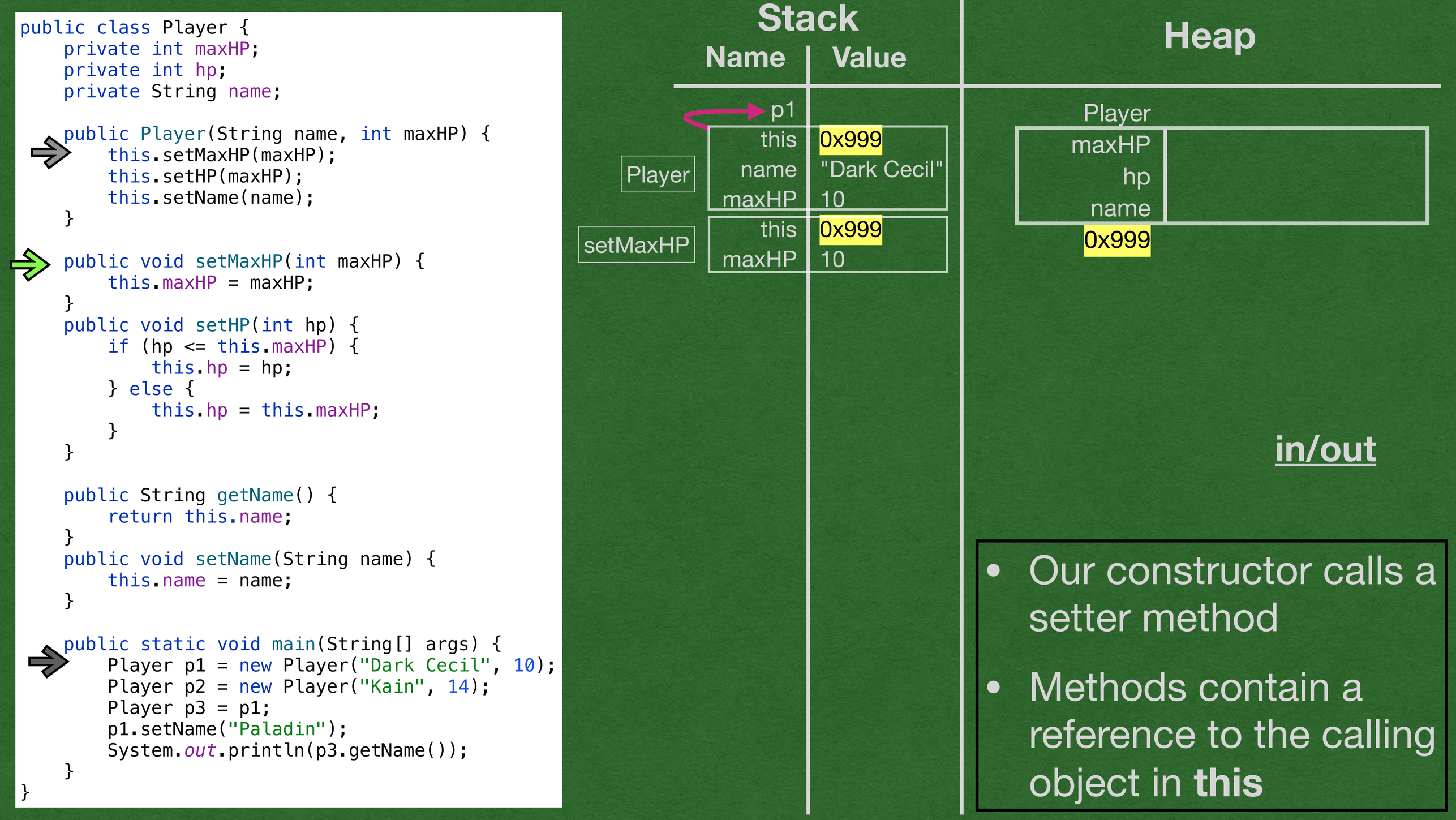
- I'll stop drawing reference arrows in slides to reduce clutter
- References will be color coded

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



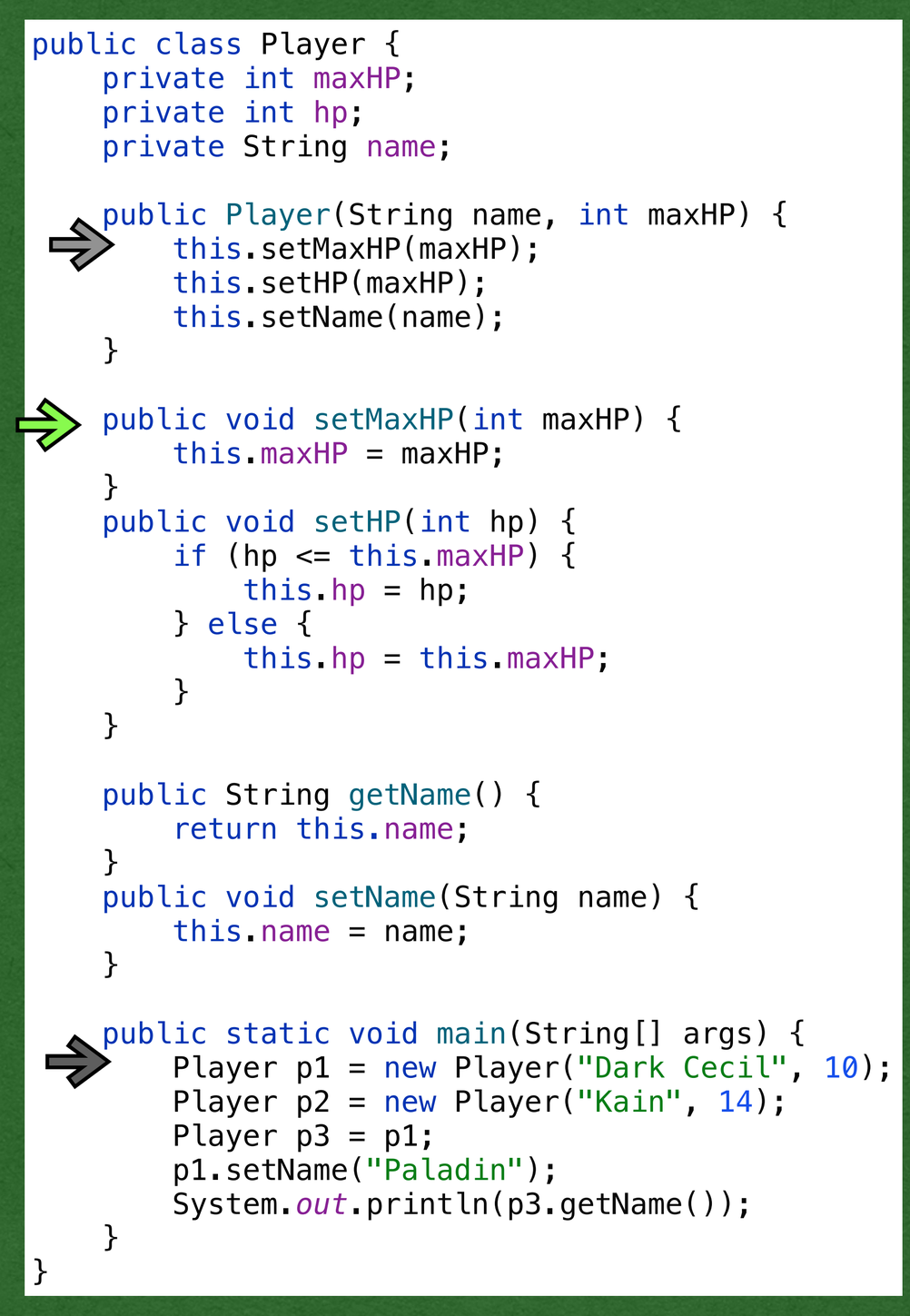
- All [non-static] methods have a "hidden" parameter of **this**
- For constructors, **this** stores a reference to the object that is being created

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



- Our constructor calls a setter method
- Methods contain a reference to the calling object in **this**

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



in/out

- This method was called by **this** in the Player constructor stack frame which stores the reference 0x999
- 0x999 is the object that called setMaxHP so that stack frame's **this** stores 0x999

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player	
this	0x999
name	"Dark Cecil"
maxHP	10
setMaxHP	
this	0x999
maxHP	10

Heap

Player

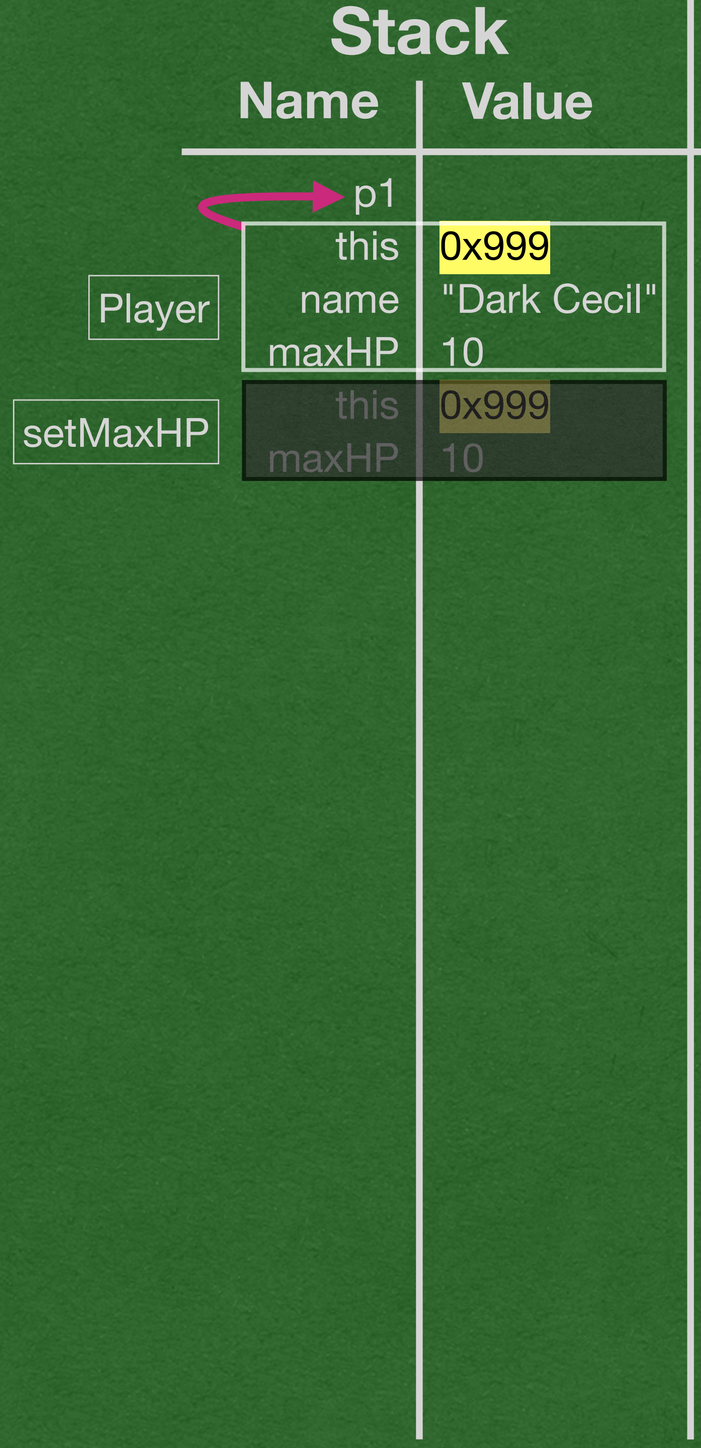
maxHP	10
hp	
name	

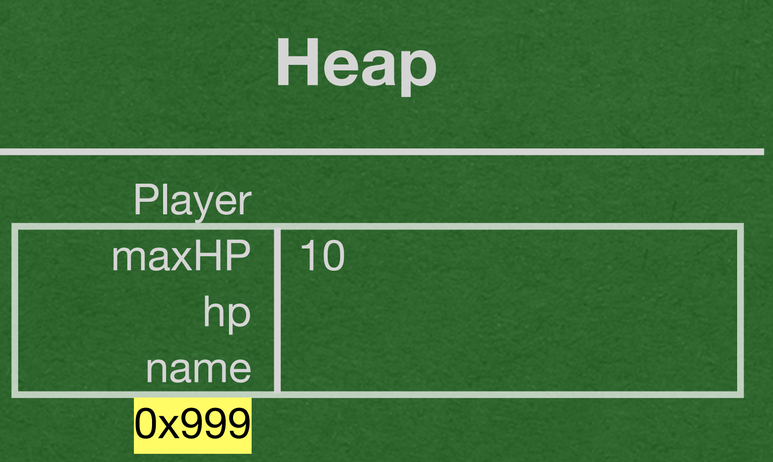
0x999

in/out

- The setter method changes the value of a variable stored in the heap
- Follow the reference stored in **this** and set its maxHP variable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```





in/out

Notation Note:

- I'll gray out a stack frame that is removed from the stack
- This will have the same meaning as crossing it out
- Makes the variables readable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player: this	0x999
Player: name	"Dark Cecil"
Player: maxHP	10
setMaxHP: this	0x999
setMaxHP: maxHP	10
setHP: this	0x999
setHP: hp	10

Heap

Player

maxHP	10
hp	10
name	

0x999

in/out

- Calling setHP will set the hp variable on the heap for this object

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



Stack

Name	Value
p1	
Player: this	0x999
Player: name	"Dark Cecil"
Player: maxHP	10
setMaxHP: this	0x999
setMaxHP: maxHP	10
setHP: this	0x999
setHP: hp	10
setName: this	0x999
setName: name	"Dark Cecil"

Heap

Player

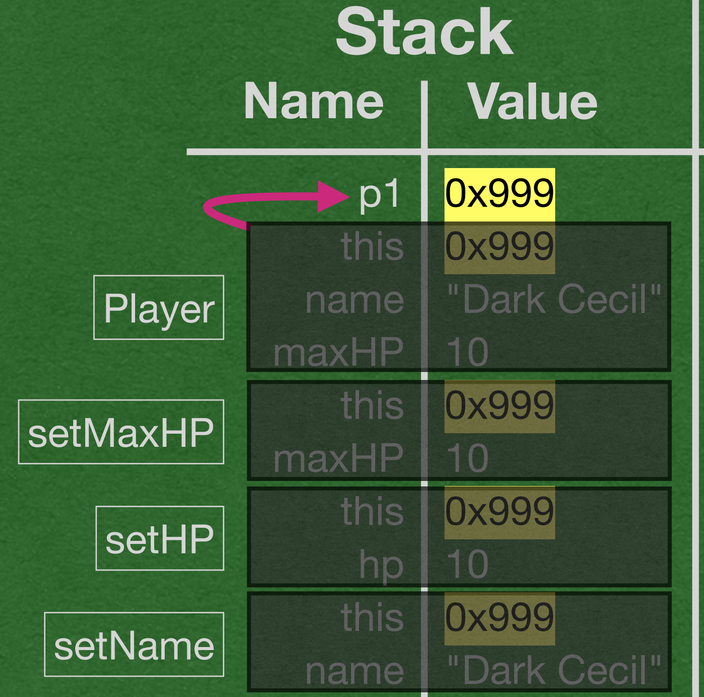
maxHP	10
hp	10
name	"Dark Cecil"

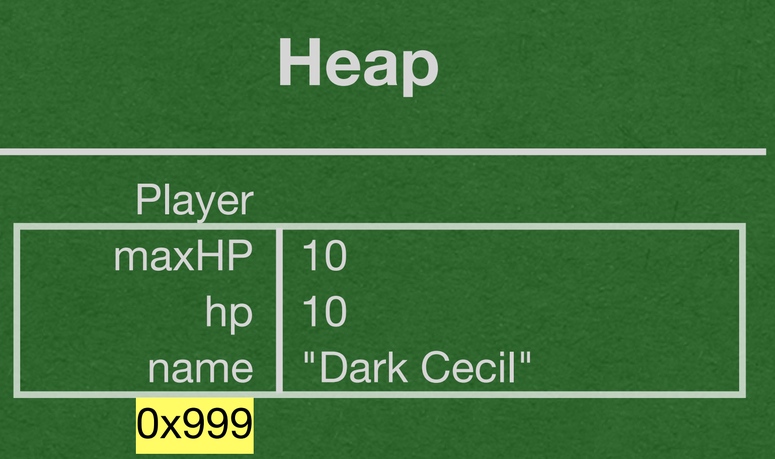
0x999

in/out

- Repeat the process for name

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

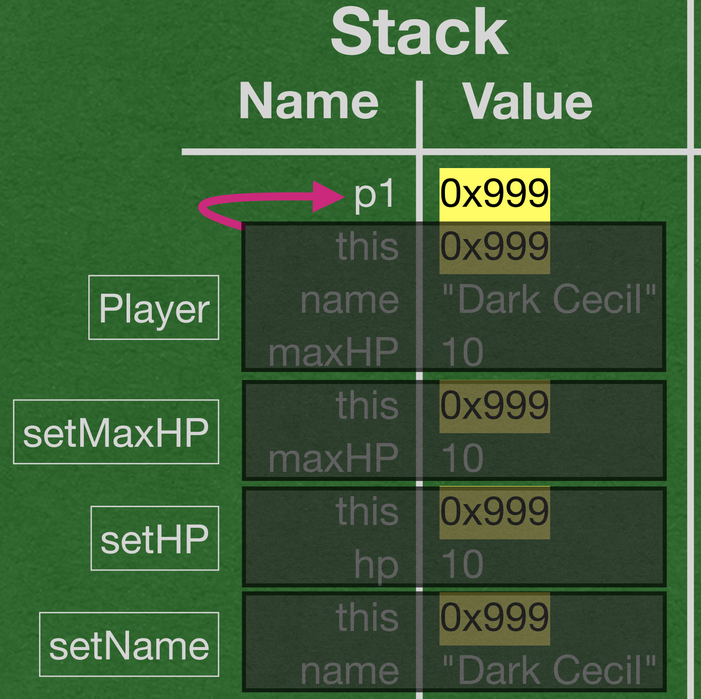


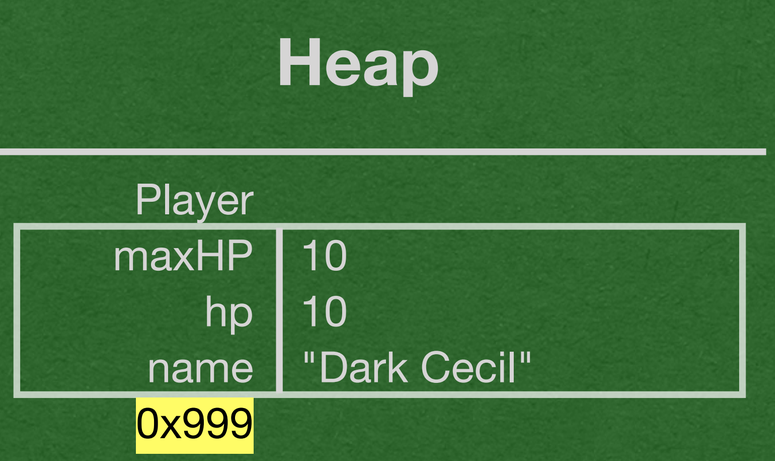


in/out

- Constructor method calls return a reference to the object that was created

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

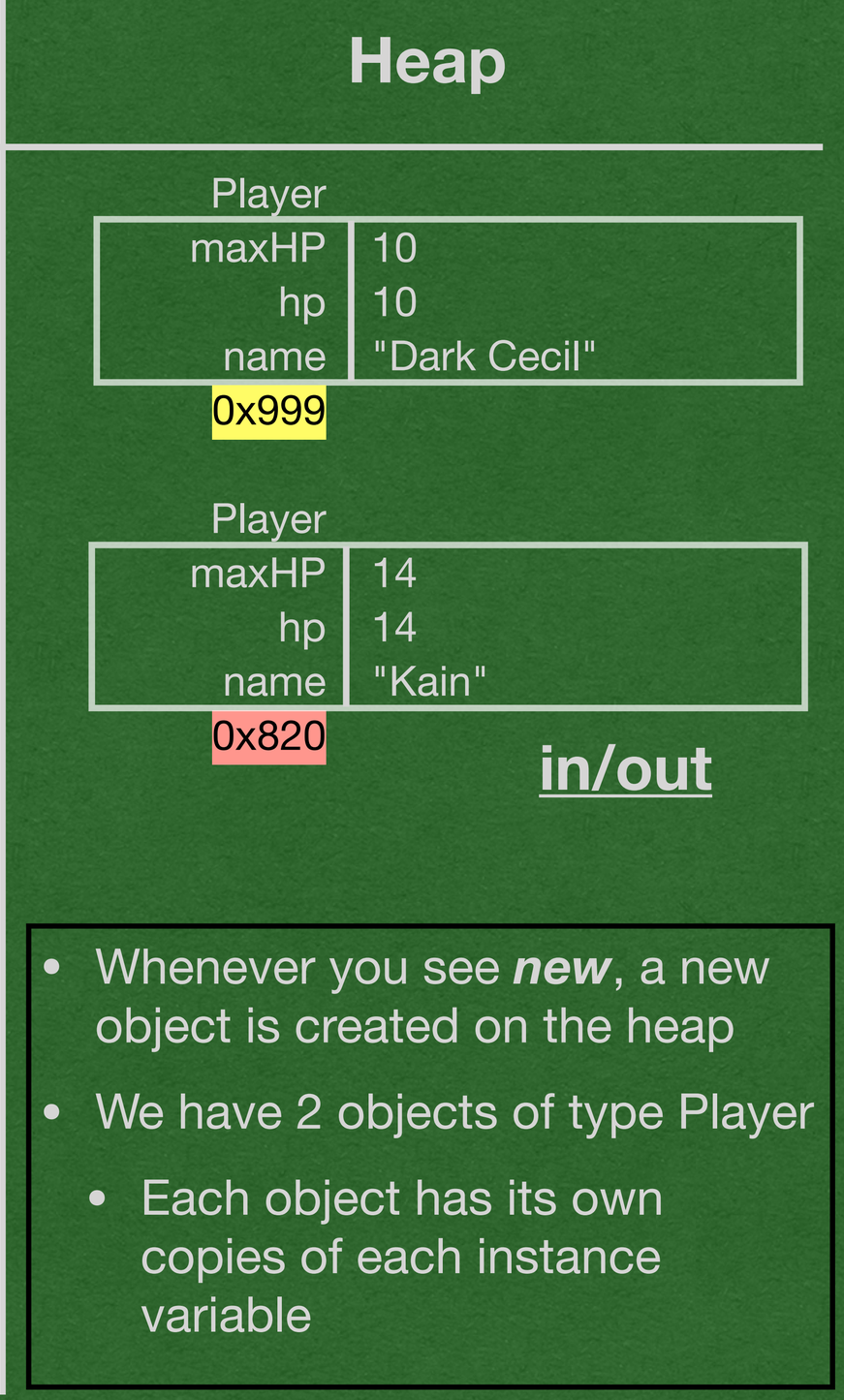




in/out

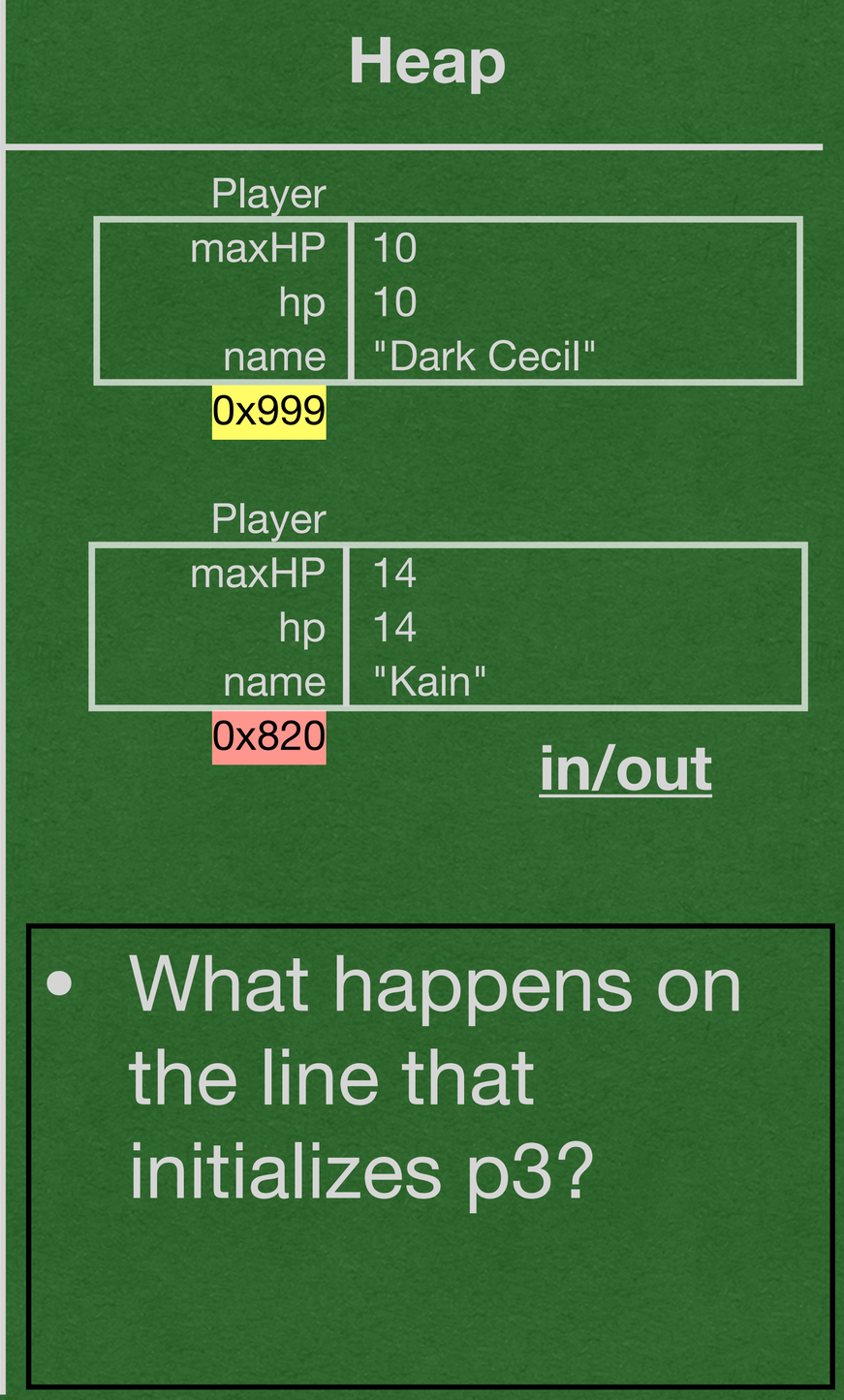
- What happens on the line that initializes p2?

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



- Whenever you see ***new***, a new object is created on the heap
- We have 2 objects of type Player
  - Each object has its own copies of each instance variable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



- What happens on the line that initializes p3?

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

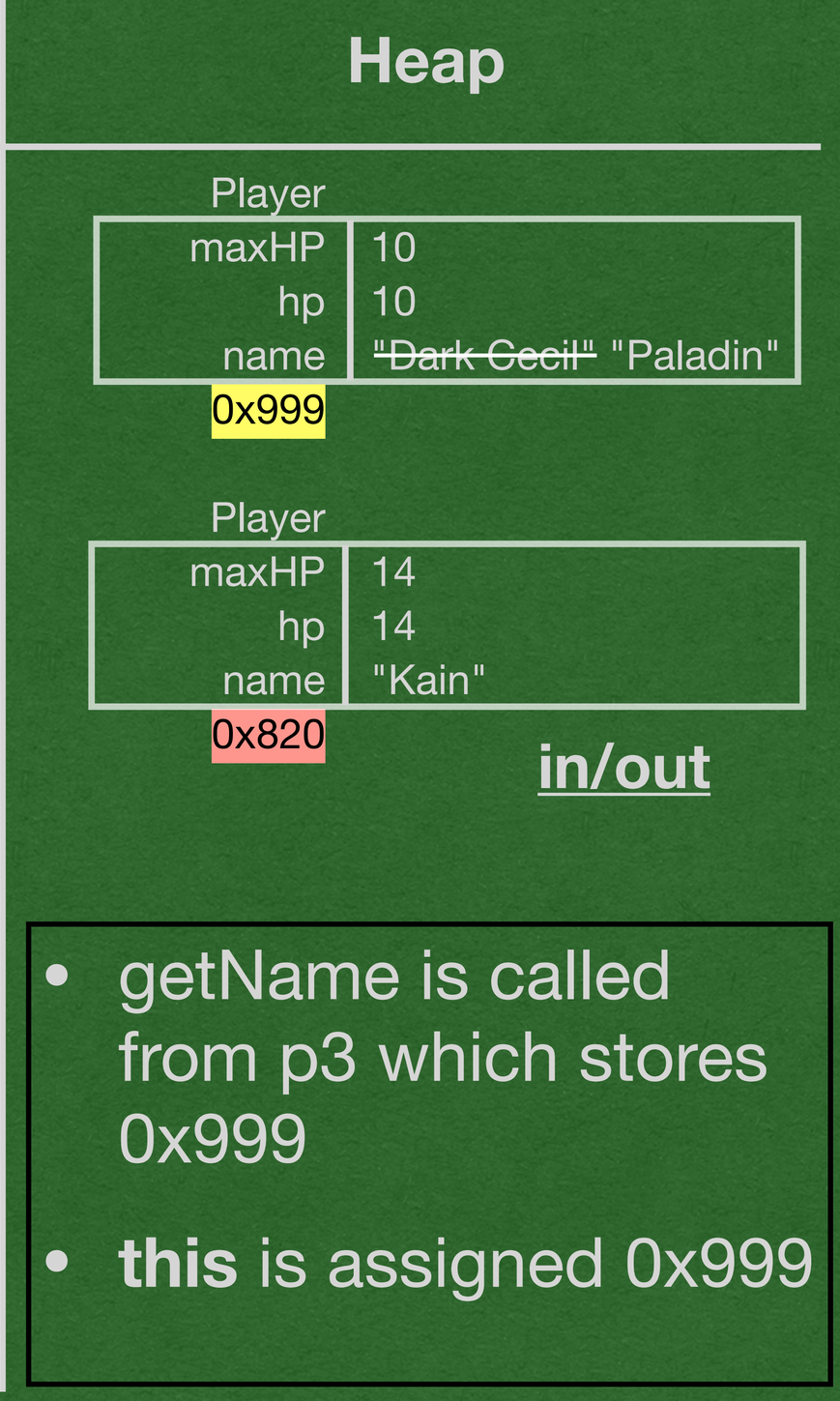
- If you **don't** see **new**, **no** object is created
- Assign p3 the same reference stored in p1
- Still only 2 objects on the heap

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

in/out

- setName is called from p1 which stores 0x999
- **this** is assigned 0x999

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



in/out

- getName is called from p3 which stores 0x999
- **this** is assigned 0x999

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

in/out

Paladin

- p1 and p3 ***refer*** to the same object
- Any change made using one variable, affects both variables!

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

in/out

Paladin

- End Program